USING SECURITY ATTACK SCENARIOS TO ANALYSE SECURITY DURING INFORMATION SYSTEMS DESIGN

Haralambos Mouratidis
School of Computing and Technology, University of East London, England
Email: h.mouratidis@uel.ac.uk

Paolo Giorgini
Department of Information and Communication Technology, University of Trento, Italy
Email: Paolo.Giorgini@dit.unitn.it

Gordon Manson
Department of Computer Science, University of Sheffield, England
Email: g.manson@uel.ac.uk

Keywords: Information Systems Analysis, Systems Engineering Methodologies, Security, Scenarios

Abstract: It has been widely argued in the literature that security concerns should be integrated with software engineering practices. However, only recently work has been initiated towards this direction. Most of this work, however, only considers how security can be analysed during the development lifecycles and not how the security of an information system can be tested during the analysis and design stages. In this paper we present results from the development of a technique, which is based on the use of scenarios, to test the reaction of an information system against potential security attacks.

1. INTRODUCTION

In previous papers, we have presented a process that allows developers to identify the security requirements of an information system (Mouratidis, 2003), reason about a suitable architectural style (Mouratidis, 2003a), and successfully transform security requirements to design (Mouratidis, 2003b).

However, an important issue is to test how the system under development copes with possible attacks. Testing is widely considered an important activity that helps to identify errors in a system and techniques such as control and data flow testing, formal specifications, special testing languages, and test tools have been used for many years in testing systems, and they are considered valuable solutions for many projects. However, most of these approaches are difficult to apply, they require special training and skills, and they employ their own concepts and notations (Ryser, 1999).

These requirements usually conflict with many of the characteristics that a security oriented process should demonstrate, such as to be clear and well guided, to allow non-security specialists to consider security issues in the development process and to employ the same concepts and notations throughout the development lifecycle of a software system.

This paper presents results from the development of a scenario-based technique to test how an information system under development copes against potential security attacks. Section 2 provides a brief overview of the Tropos methodology necessary for readers non-familiar with the methodology. Section 3 discusses our approach by describing the Security Attack Scenarios, whereas Section 4 illustrates our approach with the aid of an example taken from the health care sector. In Section 5 we present a discussion of related work and Section 6 presents some concluding remarks and future work.

2. THE TROPOS METHODOLOGY

Tropos is a development methodology tailored to describe both the organisational environment of a system and the system itself.

Tropos adopts the *i** modelling framework (Yu, 1995), which uses the concepts of actors, goals, tasks, resources and social dependencies for defining the obligations of actors (dependees) to other actors (dependers). Actors have strategic goals and intentions within the system or the organisation and represent (social) agents (organisational, human or software), roles or positions (represent a set of roles. A goal represents the strategic interests of an actor. In Tropos we differentiate between hard goals (simply called goals hereafter) and soft goals. Soft goals represent non-functional requirements and have no clear definition or criteria for deciding whether they are satisfied or not (Yu, 1995). An example of a soft goal is "the system should be scalable". A task represents a way of doing something. Thus, for example a task can be executed in order to satisfy a goal. A resource represents a physical or an informational entity while a dependency between two actors indicates that one actor depends on another to accomplish a goal, execute a task, or deliver a resource.

Therefore, in Tropos we consider the system as an actor, which can be decomposed to sub-actors, and we delegate to it goals to be satisfied (functional requirements). In order to satisfy such goals we can design the system in different ways and these can have different affect on the non-functional requirements, such as performance, reliability and security.

In previous papers (Mouratidis, 2003 – Mouratidis, 2003a – Mouratidis, 2003b), we have presented how we extended the Tropos methodology, by introducing the concepts of security reference diagram and security constraints and by redefining existing Tropos concepts such as secure entities, secure dependencies, and secure capabilities to enable it to consider security aspects throughout the whole development process.

A *security diagram* (Mouratidis, 2002) represents the connection between security features, threats, protection objectives, and security mechanisms that help towards the satisfaction of the objectives. Security features represent security related features that the system-to-be must have. Protection objectives represent a set of principles that contribute towards the achievement of the security features. Threats on the other hand represent circumstances that have the potential to cause loss or problems that can put in danger the security features of the system, while security mechanisms identify possible protection mechanisms of achieving protection objectives.

A s*ecurity constraint* (Mouratidis, 2002) is defined as a constraint that is related to the security of the system, whereas *secure entities* represent any secure goals/tasks/resources of the system. *Secure goals* are introduced to the system to help in the achievement of a *security constraint*. A *secure goal* (Mouratidis, 2002) does not particularly define how the *security constraint* can be achieved, since (as in the definition of goal) alternatives can be considered. However, this is possible through a *secure task*, since a task specifies a way of doing something (Yu, 1995). Thus, a *secure task* represents a particular way for satisfying a *secure goal*. For example, for the secure goal *Authorise Access*, we might have secure tasks such as *Check Password* or *Check Digital Signatures*. A resource that is related to a *secure entity* or a *security constraint* is considered a *secure resource*. For example, an actor depends on another actor to receive some information and this dependency (resource dependency) is restricted by a constraint *Only Encrypted Info*.

A *secure dependency* (Mouratidis, 2003) introduces *security constraint*(s), proposed either by the depender or the dependee in order to successfully satisfy the dependency. For example a *Doctor* (depender) depends on a *Patient* (dependee) to obtain *Health Information* (dependum). However, the *Patient* imposes a *security constraint* to the *Doctor* to share *health information only if consent is obtained*. Both the depender and the dependee must agree in this constraint (or constraints) for the secure dependency to be valid. That means, in the depender side, the depender expects from the dependee to satisfy the *security constraints* while in the dependee side, a secure dependency means that the dependee will make an effort to deliver the dependum by satisfying the *security constraint*(s).

A *secure capability* (Mouratidis, 2003b) represents the ability of an actor to achieve a secure goal, carry out a secure task and/or deliver a secure resource. For example, consider an actor that is responsible for providing cryptographic services in an information system. This actor should posses (amongst other) secure capabilities to *decrypt incoming data* and *encrypt outgoing data*.

3. ATTACK SCENARIOS

The popularity of scenarios have been increased among software engineers and are proven to be valuable for eliciting information about systems requirements, communicating with stakeholders and providing context for requirements (Ryser, 2000). As a result, scenarios have been applied in many different areas of computer science research, such as software engineering (Potts, 1994), business-process reengineering (Anton, 1994), and user interface design (Carroll, 1991). In particular, many cases can

be found in the literature (Ryser, 1999 – Ryser 2000 – Lalioti, 1995), where scenarios have been used for the validation of requirements.

We have decided to choose a scenario-based approach because scenarios can be easily integrated within development methodologies and can be adapted to the methodology's notation and concepts. This is due to the fact that scenarios can be represented in various ways (Ryser, 2000). In this research, a scenario, called Security Attack Scenario, is represented as an enhanced Tropos diagram, which aims to analyse how the system copes in different kinds of security attacks.

Therefore a scenario should include enough information about the system and its environment to allow validation of the system's security requirements with respect to particular attacks. As such, we define a Security Attack Scenario *as an attack situation describing the actors of a software system and their secure capabilities as well as possible attackers and their goals, and it identifies how the secure capabilities of the system's actors prevent (if they prevent) the satisfaction of the attackers' goals.*

The presented approach aims to identify the goals and the intentions of possible attackers, identify through these a set of possible attacks to the system (test cases), and apply these attacks to the system to see how it copes. By analysing the goals and the intentions of the attackers the developer obtains valuable information that helps to understand not only the *how* the attacker might attack the system, but also the *why* an attacker wants to attack the system. This leads to a better understanding of how possible attacks can be prevented. In addition, the application of a set of identified attacks to the system contributes towards the identification of attacks that the system might not be able to cope (failed test cases) and this leads to the re-definition of the actors of the system and the addition of new secure capabilities to enable them to protect against those attacks.

A Security Attack Scenario involves a possible attacker, possible attack(s), the resources that are attacked, and the actors of the system related to the attack together with their secure capabilities.

An attacker is depicted as an actor who aims to break the security of the system. The attacker intentions are modelled as goals and tasks and their analysis follows the same reasoning techniques that the Tropos methodology employs for goal and task analysis. Attacks are depicted as dash-lined links (called attack links) that contain an "attacks" tag, starting from one of the attackers goals and ending to the attacked resource.

For the purpose of a Security Attack Scenario, a differentiation takes place between internal and external actors of the system. Internal actors represent the core actors of the system whereas external actors represent actors that interact with the system. Such a differentiation is essential since it allows developers to identify different attacks to resources of the system that are exchanged between external and internal actors of the system.

The process is divided into three main stages: creation of a scenario, validation of the scenario, and testing and redefinition of the system according to the scenario. Even though the presented process is introduced as a sequence of stages, in reality is highly iterative and stages can be interchanged according to the perception of the developers. The following three sub-sections describe each of those stages.

3.1 Scenario Creation

There are two basic steps in the creation of a scenario. The first step involves the identification of the attackers' intentions and the possible attacks to the system and the second step involves identification of possible countermeasures of the system to the indicated attacks.

3.1.1 Identify the intentions of a possible attacker

During the first step, the intentions of an attacker are analysed in terms of goals and tasks. Some of these goals can be identified by the threats modelled on the security reference diagram. For example, a possible threat to a system could be the application of cryptographic attacks, i.e. attacks aiming to modify the content of messages transmitted across the network. Such a threat could introduce the goal "perform cryptographic attacks" to a potential attacker. However, other goals (apart from the ones introduced by the threats identified in the security reference diagram) could be derived from the analysis of a possible attacker's intentions. This is due to the fact that an attack is an exploitation of a system's vulnerability, whereas a threat is a circumstance that has the potential to cause loss or harm (Schneier, 2000). Therefore, an attack can lead to a threat only if the exploitation of the vulnerability leads to a threat. This means that some attacks can be successful but do not lead to threats as other system features protect the system.

When the analysis of the attacker's intentions has been completed, possible attacks to the resources of the system are indicated using attack links.

3.1.2 Identify possible countermeasures

The next step in the creation of a security attack scenario involves the identification of the actors of the system that posses capabilities to prevent the identified, from the previous step, attacks.

Secure capabilities can prevent attacks in the information system in the sense that an actor with such capabilities can react to the attacks.

Therefore, the actors (internal and external) of the system related to the identified attack(s) are modelled. The secure capabilities, of each actor, that help to prevent the identified attacks are identified and dashed-links (with the tag "help") are provided indicating the capability and the attack they help to prevent. As an example, consider an internal actor that depends on an external actor to obtain some private information. An attacker aims to read the transmitted data (eavesdropping). However, the external and the internal actors could have been assigned with secure capabilities to encrypt any data transmitted between them. As a result, eavesdropping becomes very difficult, since the data is transmitted across the network only encrypted.

3.2 Scenario Validation

When the scenarios have been created, they must be validated. Therefore, the next stage of the process involves the validation of the scenario. Software inspections are proved as effective means for document-based validation (Kosters, 2001) and as such are the choice of this research for the validation of the security attack scenarios. The inspection of the scenarios involves the identification of any possible violations of the Tropos syntax and of any possible inconsistency between the scenarios and the models of the previous stages. Such an inspection involves the use of validation checklists. Consider, for instance, the following checklist.

1. Is a name defined for each scenario?
2. Are actors represented using the correct notation?
3. Are attack links and help links correctly denoted?
4. Do the attack scenarios capture all possible attacks?
5. Do different scenarios exist for the same kind of attacks?
6. Are there any missing parts on the identified scenarios? (Any links missing or any actors missing?)
7. Are there any secure capabilities identified in the previous stages not present in the scenarios?
8. Are there any actors, identified in the previous stages, related to the attacks not present in the scenarios?
9. Are there any threats identified on the security reference diagram not present on the scenarios?
10. Are all the resources that can be attacked present in the scenarios?
11. Are the non-prevented attacks correctly marked?

It must be noticed that although inspections have been proposed by this research for the validation of the security attack scenarios, different techniques could be applied depending on the developers and the nature of the system. As an example, validation techniques to requirements specification are (apart from inspections) walkthroughs and prototyping (Kosters, 2001).

When the scenarios have been validated, the next step aims to identify test cases and test, using those test cases, the security of the system against the potential attacks. Each test case is derived from a possible attack depicted in the security attack scenarios. For each test case a precondition is necessary (the state of the system before the attack), an expected system reaction (how the system reacts in the attack), and also a discussion that forms the basis for the decision regarding the test case.

The test cases are applied and a decision is formed to whether the system can prevent the identified attacks or not. The decision whether an attack can be prevented (and in what degree) or not lies on the developer. However as an indication of the decision it must be taken into consideration that at least one secure capability must help an attack, in order for the developer to decide the attack can be prevented. Attacks that cannot be prevented are notated as solid attack links (as opposed to dashed attack links).

For each attack that it has been decided it cannot be prevented, extra capabilities must be assigned to the system to help towards the prevention of that attack. In general, the assignment of extra secure capabilities is not a unique process and depends on the perception of the developer regarding the attack dangers. However, a good approach could be to analyse the capabilities of the attacker used to perform the attack and assign the system with capabilities that can revoke the attacker's capabilities.

4. AN EXAMPLE

To illustrate our approach we apply it to a case study from the medical area. This case study is part of a real-life system, called the electronic Single

Assessment Process (eSAP), under development at the University of Sheffield (Mouratidis, 2003c). The electronic Single Assessment Process (eSAP) system is a health and social care information system for the effective care of older people. To make this example simpler and more understandable, we consider a substantial part of the eSAP system.

The application of the Security Attack Scenarios to the eSAP aims to analyse the security of the system by considering the intentions of possible attackers and the secure capabilities that have been assigned to the actors of the system and provide recommendations to improve the system's security.

As derived from the analysis of the eSAP system (Mouratidis, 2003d), the three main security features are privacy, integrity and availability. According to Stallings (Stallings, 1999), the following categories of attacks can be identified that can put in danger the above security features.

Interception, in which an unauthorised party, such as a person, a program or a computer, gains access to an asset. This is an attack on privacy.

Modification, in which an unauthorised party not only gains party to but also tampers with an asset. This is an attack on integrity.

Interruption, in which an asset of the system is destroyed or becomes unavailable or unusable. This is an attack on availability.

Due to lack of space in this paper we present only scenarios regarding interception and interruption attacks.

Let us first consider an interception attack scenario in which a possible attacker wishes to attack the privacy of the system, in other words to obtain information such as assessment information or a care plan. As identified in the analysis of the eSAP system, social engineering, password sniffing and eavesdropping are the main threats to the privacy of the system.

Therefore, the attacker's main goal can be decomposed to *Read Data* and *Get Access to the System* sub-goals as shown in Figure 1. The first sub-goal involves the attacker trying to read the data that it is transmitted to and from the eSAP system, whereas the second sub-goal involves the attacker trying to break into the system and gain access to it.

To accomplish the first sub-goal the ***Attacker*** should try to read the data transferred between the ***Social Worker*** and the ***eSAP*** system's actors such as the ***Assessment Evaluator*** and the ***Authenticator***. To accomplish the second sub-goal, the ***Attacker*** might use password sniffing or social engineering.

In the first case, the ***Attacker*** scans all the resources that flow in the network looking for passwords whereas in the case of social engineering, the ***Attacker*** tries to deceive the ***Social Worker*** in order to obtain valuable information, such as their authorisation details that will allow them to gain access to the system. Therefore, for the presented attack scenario the reaction of the system should be tested (amongst other) against three test cases, read data, password sniffing and social engineering.

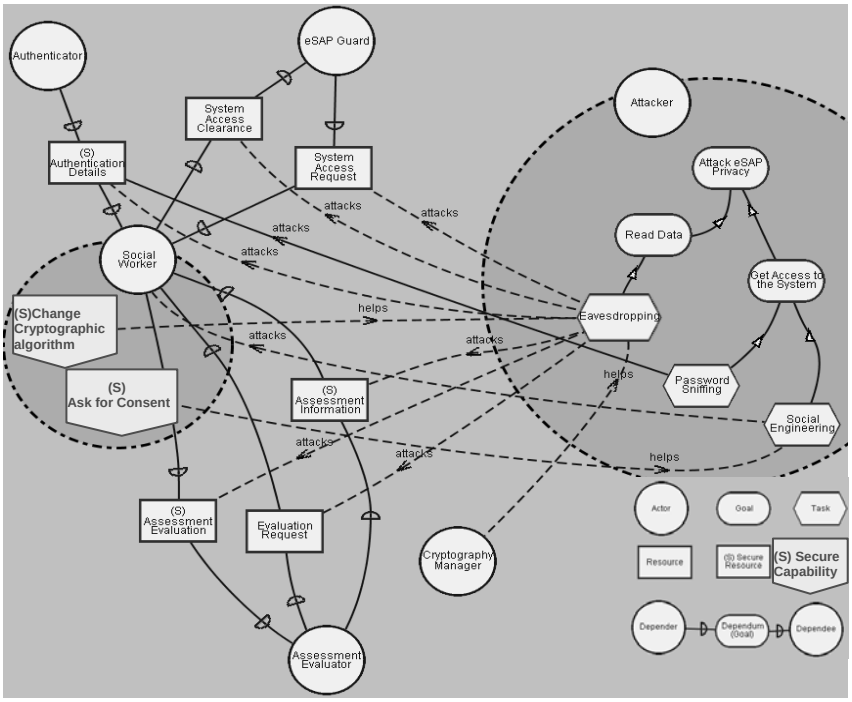

Figure 1: Interception attacks scenario

Test Case 1: read data

Precondition: The ***Social Worker*** actor tries to obtain an assessment evaluation. The ***Attacker*** tries to read the transmitted data.

System expected security reaction: The system should prevent ***Attacker*** from reading any important information.

Discussion: The ***Attacker*** will try to read the data from any resource transmitted between the external actors and the ***eSAP*** system. However, curerntly the system and its external actors have capabilities to encrypt and decrypt data. As a result all the important data is transmitted across the network encrypted and therefore it is difficult for the ***Attacker*** to read it. However, the ***Attacker*** might try to obtain (or sometimes even guess) the encryption key.

Test Case Result: The system is protected against read data attacks. However, a recommendation would be for the system to change the cryptographic algorithm often.

Test Case 2: Password sniffing

Precondition: The ***Social Worker*** tries to obtain access to the ***eSAP*** system by providing their authorisation details. The ***Attacker*** tries to intercept the authorisation details.

System expected security reaction: prevent attacker from obtaining users' passwords

Discussion: the main target of the ***Attacker*** would be all the resource transmitions between the ***Social Worker*** and the ***eSAP*** system. Currently the system does not have any kind of protection in this kind of attack. A good technique to defend against password sniffing is to use one-time-passwords. A one-time-password is a password tha is valid for only one use. After this use, it is not longer valid, and so even if the ***Attacker*** obtains such a password it is useless. However, the users must be able to gain access to the system more than once. This can be accomplished with, what is commonly known as, a password list. Each time a user tries to access the system they provide a different password from a list of passwords.

Test Case Result: Currently the system fails to protect against password sniffing attacks. For the ***eSAP*** system to be able to react in a password sniffing attack, the external actors of the system (such as the ***Nurse***, the ***Social Worker***, the ***Older Person***) must be provided with capabilities to provide passwords from a password list.

Test Case 3: Social engineering

Precondition: The ***Attacker*** tries to obtain system information directly from the ***Social Worker***.

System expected security reaction: help towards the prevention of social engineering

Discussion: The ***Attacker*** will try to deceive any external actors (such as the ***Social Worker*** in the presented scenario) into giving any confidential, private or privileged information. It is worth mentioning that the ***Attacker*** will not directly ask for this information but they will try to gain the trust of the actors and then exploit this trust.

Test Case Result: Currently the system helps towards the prevention of social engineering by requesting consent for any information to be shared. However, this alone does not guarantee the successful prevention against social engineering. A primary defence measurement against software engineering is security awareness training. Good resistance training will help to prevent actors from being persuaded to give information away.

As mentioned above, interruption attacks mainly aim the availability of the system. From an ***Attacker***'s point of view, such attacks can be mainly categorised into two main categories, physical attacks and electronic attacks (Figure 2). Physical attacks include any attacks to the infrastructure of the system, whereas electronic attacks involve attacks such as denial of service attacks.

Therefore, the ***Attacker***'s main goal (*attack eSAP availability*) can be decomposed to physical and electronic attacks.

Physical attacks involve the cutting of a communication line, or the destruction of a part of the system.

On the other hand, one of the most popular electronic attacks to the availability of a system is denial of service attacks. Since physical attacks to the eSAP system are outside the focus of this research project, only a test case involving a denial of service attack is considered.

Test Case: denial of service

Precondition: The ***Attacker*** tries to make the ***eSAP*** system unavailable by performing a denial of service attack.

System expected security reaction: the ***eSAP*** should be able to detect the attack and recover.

Discussion: During a denial of service attack, the ***Attacker*** tries to prevent the normal operation of the communication facilities of the system. Since a denial of service attack is an active attack, the main goal of the ***eSAP*** system is to detect the attack and recover from any disruption it may cause as fast as possible. Towards this direction, the actors of the system must have capabilities to operate even if some other actors have become unavailable. Mostly, denial of service attacks

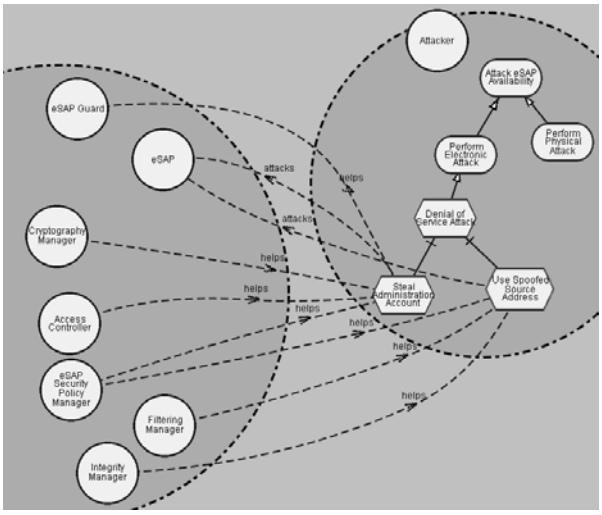

Figure 2: Interruption attacks scenario

require from ***Attackers*** to steal an administration account of a hose computer in the network. Therefore, an efficient way to prevent such attacks is to secure the system. In addition, the ***Attacker*** might make use of spoofed source address. To stop this, the system must perform filtering mainly when internal actors communicate with external ones.

Test Case Results: The ***eSAP*** system provides authorisation mechanisms and therefore helps towards the effective security of the system and in turn the prevention of denial of service attacks. However, filtering is required to make the protection against denial of service attacks even better. Therefore, an actor should be introduced to the system that will perform such filtering.

4.1 Discussion on the use of scenarios in the eSAP system

In order to test the security of the system, two different kinds of scenarios were identified involving four different test cases. By applying these test cases many useful results were obtained about the security of the eSAP system. First of all, it was identified that the system provides enough protection against some of these attacks. Secondly, for the attacks that the system did not provided adequately protection, extra actors and extra secure capabilities were identified and the following modifications (amongst other) took place in the eSAP system.

Capabilities were given to the external actors as well to the Cryptography Manager to enable them to change the cryptographic algorithm often. The lack of such capabilities was identified during the read data test case of the interception attack scenario.

The external actors of the system were given the capability to provide passwords from a password list, and the Authenticator was given capabilities to successfully process such passwords. The lack of such capabilities was identified by the application of the password-sniffing test case of the interception attack scenario.

An actor was introduced to the system to filter the eSAP in order to help towards the protection of denial of service attacks. The lack of such an actor was identified by the application of the denial of service test case of the interruption security attack scenario.

5. RELATED WORK

Scenarios have been used in many areas of software engineering, from requirements modelling (Potts, 1994) to requirements validation (Ryser, 2000).

Our idea of analysing the intentions of possible attackers is similar to the one presented by Liu et al (Liu, 2002). However, the way that our scenarios are created, verified and applied is totally different.

Yu's work is basically used to identify security requirements; the Security Attack Scenarios in our work are used to test the security requirements of the system identified in the previous development stages. So a very similar idea is applied in a different stage of the development lifecycle.

Yu argues that when the intentions of the attackers are identified we can equip the system with countermeasures, however it is never mentioned how we can do this neither provides a kind of process for providing such countermeasures.

Moreover, in our approach we consider test cases, in other words we provide ways to test each scenario for specific test cases, reason about the reaction of the system and take a final decision if the system can react to the specific attack. In cases that the system cannot react to the attack, possible countermeasures are discussed and secure capabilities are introduced to the actors of the system to satisfy them.

Yu's analysis takes place in a higher level than the one proposed by us. Yu proposes the analysis to take place during the early requirements. This could be a bit superficial for security since modelling security requirements as softgoals does not adequately model security (Mouratidis, 2002). In our case, we know the secure capabilities of the

actors of the system (and therefore we have a more precise idea of what security measurements our system has) and we can reason about the security attacks according to those capabilities.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented results from the development of a scenario-based approach to test how a software system under development copes against potential security attacks.

The introduction of security attack scenarios to test the system's response to potential attacks provides developers the ability to realistically check how the developed system will react to possible security attacks. This, in turn, allows developers to re-consider particular system functions with respect to security until the system under development satisfies all the security requirements.

The presented work is part of our efforts aiming to extend the Tropos methodology in a degree that will allow developers to successfully consider security issues during the whole development lifecycle of an information system.

Therefore, future work includes the full integration of the presented technique within the security oriented process of the Tropos methodology, and its application to more case studies in order to further assess its validity.

REFERENCES

Anton, A.I., McCracken W.M., Potts C., 1994. Goal Decomposition and Scenario Analysis in Business Process Reengineering, *Proceedings of the 6th Conference on Advanced Information Systems (CAiSE-1994)*, The Netherlands.

Carroll, J.M., Rosson, M.B., 1991. Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario, *IBM Research Report*, Human Computer Interaction, RC 17908 (75365).

Kosters, G., Pagel, B.U., Winter, M., 1997. Coupling Use Cases and Class Models, *Proceedings of the BCS-FACS/EROS workshop on "Making Object Oriented Methods More Rigorous"*, Imperial College, London-England.

Lalioti, V., Theodoulidis, C., 1995. Visual Scenarios for Validation of Requirements Specification, *Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering (SEKE'95)*, Rochville, Maryland-USA.

Liu, L., Yu, E., Mylopoulos, J., 2002. Analyzing Security Requirements as Relationships Among Strategic Actors, *Proceedings of the 2nd Symposium on Requirements Engineering for Information Security (SREIS'02)*, Raleigh-North Carolina.

Mouratidis, H., 2002. Extending Tropos Methodology to Accommodate Security, *Progress Report*, Computer Science Department, University of Sheffield.

Mouratidis, H., 2003d. Analysis and Design of a Multiagent System to Deliver the Single Assessment Process for Older People, *RANK Report*, Computer Science Department, University of Sheffield.

Mouratidis, H., Giorgini, P., Manson, G., 2003a. Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems, *Proceedings of the 15th Conference on Advance Information Systems (CAiSE-2003), Velden-Austria*.

Mouratidis, H., Giorgini, P., Manson, G., 2003b. Modelling Secure Multiagent Systems, *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, Melbourne-Australia.

Mouratidis, H., Giorgini, P., Manson, G., Gani A., 2003. Analysing Security Requirements of Information Systems Using Tropos, *Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS-2003), Angers-France.*

Mouratidis, H., Philp, I., Manson, G., 2003c. A Novel Agent-Based System to Support the Single Assessment Process of Older People, (to appear) *Journal of Health Informatics.*

Potts, C., Takahashi, K., Anton A.I., 1994. Inquiry Based Requirements Analysis, *IEEE Software*, March 1994.

Ryser, J., Glinz, M., 1999. A Practical Approach to Validating and Testing Software Systems Using Scenarios, *Proceedings of the Third International Software Quality Week Europe (QWE'99)*, Brussel-, Belgium.

Ryser, J., Glinz, M., 2000. *SCENT* - A Method Employing Scenarios to Systematically Derive Test Cases for System Test, *Technical Report 2000.03*, Institut für Informatik, University of Zurich.

Schneier, B., 2000. *Secrets and Lies: Digital Security in a Networked World*, John Willey and Sons.

Stallings, W., 1999. *Cryptography and Network Security: Principles and Practice*, Prentice-Hall, Second Edition.

Yu, E., 1995. Modelling Strategic Relationships for Process Reengineering, *PhD thesis*, Department of Computer Science, University of Toronto, Canada.